Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Криптография и информационная безопасность”

Индивидуальная работа №3  
“Алгоритм электронной подписи RSA”

Преподаватель: Cerbu Olga

Студент: Маруневич Николай

Кишинев 2022

# История создания

Название RSA состоит из начальных букв фамилий Ривест, Шамир и Адлеман, - ученых, которые впервые публично описали подобные в 1977 году. Клиффорд Кокс, английский математик, работавший на спецслужбы Великобритании, впервые разработал эквивалентную систему в 1973 году, но она не была рассекречена до 1997 г.

Идея асимметричного ключа криптосистемы приписывается Диффи и Хеллману, которые опубликовали концепцию в 1976 году, представив цифровые подписи и попытавшись применить теорию чисел. Их формулировка использует общий секретный ключ, созданный из экспоненциации некоторого числа по модулю простого числа. Тем не менее, они оставили открытой проблему реализации этой функции, поскольку принципы факторинга не были хорошо изучены в то время.

Ривест, Ади Шамир и Адлеман в Массачусетском технологическом институте предприняли несколько попыток в течение года, чтобы создать однонаправленную функцию, которую трудно раскодировать. Ривест и Шамир (как компьютерные ученые) предложили множество потенциальных функций, в то время как Адлеманом (как математиком) осуществлялся поиск «слабых мест» алгоритма. Они использовали много подходов и в конечном итоге в апреле 1977 года разработали окончательно систему, сегодня известную как RSA.

# Алгоритм RSA

RSA относится к асимметричным алгоритмам шифрования: если для шифрования используется открытый ключ, то для дешифрования используется закрытый, и наоборот.

Первое свойство позволяет кому угодно зашифровать сообщение открытым ключом в адрес владельца закрытого ключа и тем самым обеспечить его конфиденциальность.

Второе свойство позволяет владельцу ключа зашифровать хэш сообщения закрытым ключом, чтобы кто угодно мог расшифровать зашифрованный хэш, сравнить его с хэшем сообщения и определить, было ли сообщение модифицировано.

## Генерация ключей

1. Первый этап генерации ключей — случайный выбор двух достаточно больших простых чисел p и q. Натуральное число x называется простым, если у него есть ровно два различных делителя: 1 и x. Все другие делители числа располагаются на отрезке от 2 до квадратного корня из x, однако достаточно проверять на кратность только простые делители, принадлежащие этому отрезку. Также эти два числа должны быть одинакового размера (например по 256 бит каждое)
2. Далее вычисляется произведение этих простых чисел - модуль, или *n*. Модуль является первым элементом открытого и закрытого ключей. В реализациях RSA, используемых для защиты информации, используются алгоритмы поиска простых чисел, позволяющие найти простые числа с большим числом разрядов; благодаря тому, что лучший известный алгоритм разложения числа на простые множители работает за время, пропорциональное экспоненте от количества разрядов, считается что восстановить пару простых чисел по рассматриваемому элементу открытого ключа невозможно.
3. Вычисляется значение функции Эйлера от числа *n*. Функция Эйлера от x равна количеству натуральных чисел, не больших x и взаимно простых с ним. Для n это количество будет равно произведению p-1 и q-1. полученное значение будет вторым элементом закрытого и открытого ключей.
4. Далее выбирается публичная экспонента *е*. Это целое взаимно простое со значением функции φ(n) число. Обычно в качестве *e* берут простые числа, содержащие небольшое количество единичных бит в двоичной записи.
5. Затем вычисляется секретная экспонента *d,* которая должна удовлетворять условию



Таким образом, нашим публичным ключом будет пара {*e, n*}, а приватным ключом - {*d, n*}

## Шифрование

Допустим, Боб хочет отправить сообщение Алисе. Алиса тогда вкладывает в общий доступ публичный ключ {*e, n*}. Затем Боб берет незашифрованное сообщение и, используя следующую функцию, создает зашифрованное сообщение *с*:



Зашифрованное сообщение Боб посылает Алисе

## Расшифровка

С помощью приватного ключа {*d, n*} Алиса расшифровывает сообщение с.



При этом публичный ключ доступен всем, кто хотел бы написать Алисе, а приватный ключ доступен только самой Алисе, поэтому расшифровать сообщение может только она.

# Цифровая подпись

Цифровая подпись - это реквизит электронного документа, который позволяет проверить авторство сообщения и отсутствие искажений информации.

Широко применяемая в настоящее время технология электронной подписи основана на асимметричном шифровании с открытым ключом и опирается на следующие принципы:

* Можно сгенерировать пару очень больших чисел (открытый ключ и закрытый ключ) так, чтобы, зная открытый ключ, нельзя было вычислить закрытый ключ за разумный срок. Механизм генерации ключей строго определён и является общеизвестным. При этом каждому открытому ключу соответствует определённый закрытый ключ. Если, например, Иван Иванов публикует свой открытый ключ, то можно быть уверенным, что соответствующий закрытый ключ есть только у него.
* Имеются надёжные методы шифрования, позволяющие зашифровать сообщение закрытым ключом так, чтобы расшифровать его можно было только открытым ключом. Механизм шифрования является общеизвестным.
* Если электронный документ поддается расшифровке с помощью открытого ключа, то можно быть уверенным, что он был зашифрован с помощью уникального закрытого ключа. Если документ расшифрован с помощью открытого ключа Ивана Иванова, то это подтверждает его авторство: зашифровать данный документ мог только Иванов, т.к. он является единственным обладателем закрытого ключа.

Однако шифровать весь документ было бы неудобно, поэтому шифруется только его хеш — небольшой объем данных, жёстко привязанный к документу с помощью математических преобразований и идентифицирующий его. Зашифрованный хеш и является электронной подписью.

## Отправка сообщения

Открытый текст m подписывается с помощью подписи s и приватного ключа автора сообщения {d, n}:



Пара {m, s} передается получателю

## Прием сообщения

Получатель вычисляет прообраз сообщения из подписи {m, s} в помощью публичного ключа {e, n}



Далее m и m’ сравниваются, и если они совпадают, авторство подтверждено.

# Безопасность

Для работы RSA требуется выбрать довольно много параметров. К сожалению, невинные на первый взгляд методы их выбора могут навредить безопасности.

Разработчики несут ответственность за выбор простых чисел p и q, составляющих модуль RSA. Этот процесс чрезвычайно медленный по сравнению с генерацией ключей для других криптографических протоколов, где достаточно просто выбрать несколько случайных байтов. Поэтому, вместо того чтобы генерировать действительно случайное простое число, разработчики часто пытаются создавать числа определенной формы. Это почти всегда плохо кончается.Более того, p и q должны быть выбраны независимо друг от друга. Если p и q совместно используют приблизительно половину своих старших битов, то N может быть вычислено с использованием метода Ферма.

Поскольку использование закрытого ключа большого размера отрицательно влияет на время расшифровки и подписи, у разработчиков есть стимул выбирать небольшую d, особенно в случаях устройств с низким потреблением энергии, таком как смарт-карты. Тем не менее, злоумышленник может восстановить закрытый ключ, когда d меньше корня 4-й степени из N. Вместо этого разработчикам стоит выбирать большое значение d. Однако сложность этого подхода увеличивает вероятность незначительных ошибок реализации, которые могут привести к восстановлению ключа.

Как и в случае c секретной экспонентой, при выборе значения е разработчики хотят использовать небольшие открытые экспоненты, чтобы сэкономить на шифровании и проверке подписей. Обычно в этом контексте используются простые числа Ферма, в частности e = 3, 17 и 65537.

Несмотря на то, что криптографы рекомендуют использовать 65537, разработчики часто выбирают e = 3, что приводит к множеству уязвимостей в криптосистеме RSA